Technical brochure — cryptographic means, Scantic

Encryption description for regulatory filings · version 1.0 · 31 July 2026

1. Scope

Scantic is an iOS application for 3D scanning. All processing — keyframe extraction, camera pose estimation and training of the 3D model — runs entirely on the device. Cryptography is used in one optional feature only: sharing a finished scene through a secret link.

> No proprietary algorithms are used. Every primitive comes from Apple's operating system libraries: **CryptoKit** and **CommonCrypto**.

2. Algorithms and key lengths

Algorithm	Mode / parameter	Key size	Purpose
AES	GCM, 96-bit nonce	256 bits	Confidentiality and integrity of shared content
HKDF-SHA256	128-bit salt, context label	256 bits	Derivation of content keys
PBKDF2-HMAC-SHA256	600,000 iterations, 128-bit salt	256 bits	Derivation from a user password
SHA-256	—	—	Hashing (HKDF, HMAC, server-side token digest)
TLS	1.2 / 1.3	as provided by the OS	Transport security (supplied by iOS)

3. Key derivation chain

1. A 256-bit link key `K_link` is drawn at random on the device (`SecRandomCopyBytes`).
2. For a password-protected link, a key `K_pw` is derived with PBKDF2-HMAC-SHA256, 600,000 iterations, 128-bit random salt.
3. A master key is derived with HKDF-SHA256 from `K_link` — and from `K_pw` where applicable — using a 128-bit random salt.
4. Two distinct 256-bit keys are derived from it with HKDF-SHA256 under different context labels: one for the 3D scene, one for the metadata. A key for one blob therefore cannot decrypt the other.
5. Each blob is sealed separately with AES-256-GCM.

4. Format of the encrypted data

Every sealed blob starts with a 20-byte header, authenticated as associated data (AAD): magic `SCE1` (4 bytes), format version (1), cipher identifier (1), reserved (2), nonce (12). The ciphertext and the GCM authentication tag follow.

Pre-processing

Before encryption the 3D scene is converted from PLY to SPZ (quantisation plus gzip), which reduces its size by roughly a factor of six. Metadata (scene name, licence, initial camera pose) is serialised as JSON. No other pre-processing is applied.

Post-processing

None. The sealed blob is transmitted to the server as-is over HTTPS.

5. Key management

The link key is never sent to the server. It travels in the **fragment** of the share URL:

`https://scantic.app/s/<token>#k=<base64url key>`

By construction browsers do not transmit the fragment in requests: it appears neither in server logs nor at any intermediary. Decryption happens in the recipient's browser. The supplier therefore cannot read private or password-protected shares.

The token in the URL path is stored server-side only as a SHA-256 digest. A password, where used, is never transmitted or stored: only the key derived from it is used, locally. No keys are retained on the device beyond what sharing requires; the management token that allows a link to be revoked is kept in the system keychain.

6. Public mode

A third mode exists and is chosen explicitly by the user: publishing in the clear. It applies no content encryption, and the interface states this unambiguously before publication.

7. Protection of the process

The application ships as a binary signed by the supplier and verified by the operating system; iOS refuses to execute a modified binary. Cryptographic parameters are fixed in code and are not exposed through any setting, interface or configuration file.

8. Implementation references

The implementation lives in a single source file, `ShareCrypto.swift`, covered by an automated test harness whose vectors were independently confirmed by three implementations (CryptoKit, the Python standard library, and the browser's WebCrypto). The wire format is specified in a single binding document applied to the application, the server and the web viewer alike.

Sebastian Beyer · Effingergasse 18, 1160 Vienna, Austria · support@scantic.app